```
# Copyright 2014 The Kubernetes Authors All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

. /etc/sysconfig/heat-params

set -x
set -o errexit
set -o nounset
set -o pipefail

ssh_cmd="ssh -F /srv/magnum/.ssh/config root@localhost"

if [ "$TLS_DISABLED" == "True" ]; then
    exit 0
fi

if [ "$VERIFY_CA" == "True" ]; then
    VERIFY_CA=""
else
    VERIFY_CA="-k"
fi

if [ -z "${KUBE_NODE_IP}" ]; then
    KUBE_NODE_IP=$(curl -s http://169.254.169.254/latest/meta-data/local-ipv4)
fi

sans="IP:${KUBE_NODE_IP}"

if [ -z "${KUBE_NODE_PUBLIC_IP}" ]; then
    KUBE_NODE_PUBLIC_IP=$(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)
fi

if [ -n "${KUBE_NODE_PUBLIC_IP}" ]; then
    sans="${sans},IP:${KUBE_NODE_PUBLIC_IP}"
fi

if [ "${KUBE_NODE_PUBLIC_IP}" != "${KUBE_API_PUBLIC_ADDRESS}" ] \
        && [ -n "${KUBE_API_PUBLIC_ADDRESS}" ]; then
    sans="${sans},IP:${KUBE_API_PUBLIC_ADDRESS}"
fi

if [ "${KUBE_NODE_IP}" != "${KUBE_API_PRIVATE_ADDRESS}" ] \
```

```
        && [ -n "${KUBE_API_PRIVATE_ADDRESS}" ]; then
    sans="${sans},IP:${KUBE_API_PRIVATE_ADDRESS}"
fi

MASTER_HOSTNAME=${MASTER_HOSTNAME:-}
if [ -n "${MASTER_HOSTNAME}" ]; then
    sans="${sans},DNS:${MASTER_HOSTNAME}"
fi

if [ -n "${ETCD_LB_VIP}" ]; then
    sans="${sans},IP:${ETCD_LB_VIP}"
fi

sans="${sans},IP:127.0.0.1"

KUBE_SERVICE_IP=$(echo  $PORTAL_NETWORK_CIDR  |  awk  'BEGIN{FS="[./]";  OFS="."}{print
$1,$2,$3,$4 + 1}')

sans="${sans},IP:${KUBE_SERVICE_IP}"

sans="${sans},DNS:kubernetes,DNS:kubernetes.default,DNS:kubernetes.default.svc,DNS:kuber
netes.default.svc.cluster.local"

echo "sans is ${sans}"
cert_dir=/etc/kubernetes/certs
mkdir -p "$cert_dir"
CA_CERT=$cert_dir/ca.crt

function generate_certificates {
    _CERT=$cert_dir/${1}.crt
    _CSR=$cert_dir/${1}.csr
    _KEY=$cert_dir/${1}.key
    _CONF=$2
    _CA_CERT_TYPE=$3

    #Get a token by user credentials and trust
    auth_json=$(cat << EOF
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "id": "$TRUSTEE_USER_ID",
                    "password": "$TRUSTEE_PASSWORD"
                }
            }
        },
        "scope": {
            "OS-TRUST:trust": {
                "id": "$TRUST_ID"
```

```
            }
        }
    }
}
EOF
)

    content_type='Content-Type: application/json'
    url="$AUTH_URL/auth/tokens"
    USER_TOKEN=`curl $VERIFY_CA -s -i -X POST -H "$content_type" -d "$auth_json" $url \
        | grep -i X-Subject-Token | awk '{print $2}' | tr -d '[[:space:]]'`

       # Get CA certificate for this cluster. If the CA_CERT_TYPE is not etcd or
front-proxy, it will return the default CA cert for kubelet
    curl $VERIFY_CA -X GET \
        -H "X-Auth-Token: $USER_TOKEN" \
        -H "OpenStack-API-Version: container-infra latest" \
         $MAGNUM_URL/certificates/$CLUSTER_UUID"?ca_cert_type="${_CA_CERT_TYPE} | python
-c 'import sys, json; print(json.load(sys.stdin)["pem"])' > ${CA_CERT}

    # Generate server's private key and csr
    $ssh_cmd openssl genrsa -out "${_KEY}" 4096
    chmod 400 "${_KEY}"
    $ssh_cmd openssl req -new -days 1000 \
            -key "${_KEY}" \
            -out "${_CSR}" \
            -reqexts req_ext \
            -config "${_CONF}"

    # Send csr to Magnum to have it signed
                csr_req=$(python    -c    "import    json;    fp    =    open('${_CSR}');
print(json.dumps({'ca_cert_type':   '$_CA_CERT_TYPE',   'cluster_uuid':   '$CLUSTER_UUID',
'csr': fp.read()})); fp.close()")
    curl $VERIFY_CA -X POST \
        -H "X-Auth-Token: $USER_TOKEN" \
        -H "OpenStack-API-Version: container-infra latest" \
        -H "Content-Type: application/json" \
        -d "$csr_req" \
                      $MAGNUM_URL/certificates   |   python   -c   'import   sys,   json;
print(json.load(sys.stdin)["pem"])' > ${_CERT}
}

# Create config for server's csr
cat > ${cert_dir}/server.conf <<EOF
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = kubernetes
[req_ext]
subjectAltName = ${sans}
extendedKeyUsage = clientAuth,serverAuth
```

```
EOF

#Kubelet Certs
cat > ${cert_dir}/kubelet.conf <<EOF
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = system:node:${INSTANCE_NAME}
O=system:nodes
OU=OpenStack/Magnum
C=US
ST=TX
L=Austin
[req_ext]
subjectAltName = ${sans}
keyUsage=critical,digitalSignature,keyEncipherment
extendedKeyUsage=clientAuth,serverAuth
EOF

#admin Certs
cat > ${cert_dir}/admin.conf <<EOF
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = admin
O = system:masters
OU=OpenStack/Magnum
C=US
ST=TX
L=Austin
[req_ext]
extendedKeyUsage= clientAuth
EOF

generate_certificates server ${cert_dir}/server.conf kubelet
generate_certificates kubelet ${cert_dir}/kubelet.conf kubelet
generate_certificates admin ${cert_dir}/admin.conf kubelet

# Generate service account key and private key
echo -e "${KUBE_SERVICE_ACCOUNT_KEY}" > ${cert_dir}/service_account.key
echo -e "${KUBE_SERVICE_ACCOUNT_PRIVATE_KEY}" > ${cert_dir}/service_account_private.key

# Common certs and key are created for both etcd and kubernetes services.
# Both etcd and kube user should have permission to access the certs and key.
if [ -z "`cat /etc/group | grep kube_etcd`" ]; then
    $ssh_cmd groupadd kube_etcd
    $ssh_cmd usermod -a -G kube_etcd etcd
    $ssh_cmd usermod -a -G kube_etcd kube
    $ssh_cmd chmod 550 "${cert_dir}"
```

```
    $ssh_cmd chown -R kube:kube_etcd "${cert_dir}"
    $ssh_cmd chmod 440 "$cert_dir/server.key"
fi

# Create certs for etcd
cert_dir=/etc/etcd/certs
$ssh_cmd mkdir -p "$cert_dir"
CA_CERT=${cert_dir}/ca.crt

cat > ${cert_dir}/server.conf <<EOF
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = etcd
[req_ext]
subjectAltName = ${sans}
extendedKeyUsage = clientAuth,serverAuth
EOF

generate_certificates server ${cert_dir}/server.conf etcd
generate_certificates admin ${cert_dir}/server.conf etcd

if [ -z "`cat /etc/group | grep kube_etcd`" ]; then
    $ssh_cmd chown -R etcd:kube_etcd "${cert_dir}"
fi

# Create certs for front-proxy
cert_dir=/etc/kubernetes/certs/front-proxy
$ssh_cmd mkdir -p "$cert_dir"
CA_CERT=${cert_dir}/ca.crt

cat > ${cert_dir}/server.conf <<EOF
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = front-proxy
[req_ext]
subjectAltName = ${sans}
extendedKeyUsage = clientAuth,serverAuth
EOF

generate_certificates server ${cert_dir}/server.conf front-proxy
generate_certificates admin ${cert_dir}/server.conf front-proxy

if [ -z "`cat /etc/group | grep kube_etcd`" ]; then
    $ssh_cmd chown -R kube:kube_etcd "${cert_dir}"
fi
```